



Bug Tracking System

Documentation Report

Submitted by :-

- Ankita Jaiswal
- Prathijna Shetty
- Diksha Trivedi
- Karan
- Rishi
- Sayan Bakshi
- Shatakshi Singh
- Tushar Sachdeva



Abstract

This project aims to develop a tracking system useful for applications deployed in an organization. The bug tracking system is an application that can be accessed throughout the organization. this system can be used for logging defects against an application/module, assigning defects to individuals and tracking the defects to resolution. bug tracking is the process of finding defects in a product (testing),and making new versions of the product that fix the defects. defects are managing, evaluating and prioritizing these defects is a difficult task: bug tracking systems are computer database systems that store defects and help people to manage them.

Introduction

In the fast-paced world of software development, managing and resolving bugs efficiently is a critical aspect of delivering high-quality software products. Today, we are excited to introduce our bug tracking system, a powerful solution designed to streamline the issue management process. In software development, bugs and issues are inevitable. they can impact product quality, customer satisfaction, and project timelines. our bug tracking system is a comprehensive tool that facilitates the reporting, tracking, and resolution of software bugs within project teams. by centralizing bug-related information and providing role-based access, our system empowers project managers, developers, and testers to collaborate effectively in identifying, addressing, and closing issues.

Modules & Description of Bug Tracking System

Users Module

Purpose:

The Users module manages user-related functionalities and data. It deals with user authentication, profiles, and permissions.

Key Functionalities:

User Registration and Authentication: Allows users to create accounts and log in securely.

User Profiles: Manages user profiles, including personal information, roles, and permissions.

Permissions and Roles: Defines user roles (admin, manager, developer, tester) and their respective permissions within the system.

Use Case:

When a user logs in, the Users module ensures they have the appropriate permissions and displays content accordingly.

Projects Module

Purpose:

The Projects module is responsible for organizing software development projects and their associated information.

Key Functionalities:

Project Creation and Management: Allows users to create, update, and delete projects.

Project Details: Stores information about each project, such as name, description, start date, and end date.

Project Members: Manages the list of team members associated with a project.

Use Case:

When a user logs in, they can view a list of projects they are involved in and access project-specific information.

Teams Module

Purpose:

The Teams module focuses on organizing development teams and their roles within different projects.

Key Functionalities:

Team Creation and Management: Allows the creation and administration of development teams.

Team Members: Manages team members, assigns roles, and tracks team activity.

Use Case:

Managers use this module to assign development teams to specific projects and allocate bug-fixing responsibilities.

Bugs Module

Purpose:

The Bugs module is at the core of the Bug Tracking System and deals with the identification, reporting, and resolution of software bugs.

Key Functionalities:

Bug Reporting: Allows users to report new bugs, specifying details like title, description, and steps to reproduce.

Bug Assignment: Managers can assign bugs to development teams or individual developers.

Bug Tracking: Tracks the status of each bug (e.g., open, in progress, resolved) and provides a history of changes.

Bug Comments: Enables communication and collaboration among team members by allowing comments on each bug.

Use Case:

Developers, testers, and managers use this module to report, track, and resolve bugs during the software development process.



Technology Used

Frontend Technologies

HTML:

It is used to create the structure of the Bug Tracking System's user interface (UI). It defines the layout, headings, forms, buttons, and other UI elements.

CSS:

It is applied to the HTML elements to style the Bug Tracking System's UI. It controls the colors, fonts, spacing, and overall visual design.

JavaScript:

JavaScript adds interactivity to the Bug Tracking System. It is used for client-side validation, handling user actions (e.g., clicking buttons), and making asynchronous requests to the backend for real-time updates.

Backend Technologies

Java

Java is used to implement the server-side logic of the Bug Tracking System. It handles HTTP requests from the frontend, processes bug reports, manages user authentication, and communicates with the database.



JDBC (Java Database Connectivity):

JDBC is used to connect the Java backend to the SQL database where bug data is stored. It facilitates database operations such as inserting bug reports, updating bug statuses, and querying bug information.

SQL:

SQL defines the structure of the database tables used to store bug-related data. It enables the retrieval of bug information, filtering, and generating reports.

Project Flow

1. Project Initiation:

Project Kickoff: Gather stakeholders, including project managers, developers, testers, and other relevant team members, to discuss the need for a Bug Tracking System (BTS).

Requirements Gathering: Understand the specific needs and requirements of the organization for bug tracking and management.

2. System Design and Planning:

System Architecture: Design the architecture of the BTS, including the database structure, user interfaces, and integration points with other tools (if any).

Feature Specification: Define the key features and functionalities of the BTS, such as bug reporting, issue prioritization, user roles, notifications, and reporting.

Technology Stack: Select the technology stack and development tools that will be used to build the system.

Project Plan: Create a project plan outlining the development phases, milestones, and timelines.

3. Development:

Frontend Development: Design and develop the user interfaces for bug reporting, issue tracking, and reporting.


Backend Development: Implement the core functionality of the BTS, including bug storage, assignment, notifications, and reporting.

Database Setup: Create and configure the database for storing bug-related data.

Integration: Integrate the BTS with other tools and systems used in the organization's software development workflow (e.g., version control, project management tools).

4. Testing:

Unit Testing: Perform unit testing of individual components to ensure they work as expected.



Integration Testing: Test the interactions and integrations between different modules of the BTS.

User Acceptance Testing (UAT): Involve end-users, including developers, testers, and project managers, in UAT to validate the system's functionality and usability.

5. Deployment:

Staging Environment: Deploy the BTS to a staging environment for final testing and validation.

Production Deployment: Once the system passes all tests, deploy it to the production environment for everyday use.

6. User Training:

Training Sessions: Conduct training sessions for all users to familiarize them with the BTS. Provide documentation and user guides for reference.

7. Rollout:

Gradual Rollout: Gradually introduce the BTS to different teams or projects within the organization to ensure a smooth transition.

Feedback Collection: Gather feedback from users during the initial rollout phase and make necessary adjustments.

8. Ongoing Maintenance and Support:

Bug Fixing: Continuously monitor and address any bugs or issues that arise in the BTS.

Updates and Enhancements: Implement updates and enhancements based on user feedback and changing requirements.

9. Reporting and Analytics:

Generate Reports: Utilize the reporting and analytics features of the BTS to generate insights and reports on bug trends, resolution times, and team performance.

10. User Adoption:

Promote Adoption: Encourage all team members to consistently use the BTS for bug tracking and reporting.

Monitor Adoption: Monitor user adoption rates and address any adoption challenges.

11. Evaluation and Optimization:

Performance Evaluation: Regularly evaluate the performance of the BTS against predefined metrics and KPIs.

Optimization: Make continuous improvements to optimize the BTS for better bug tracking and management.

12. Scaling:

Scalability: Ensure that the BTS can scale as the organization grows, accommodating more users and projects.

13. End of Life (if applicable):

Retirement: Plan for the eventual retirement of the BTS when it is no longer in use or has been replaced by a more advanced system.

14. Documentation:

Documentation Maintenance: Keep documentation up to date with any changes or enhancements made to the BTS.

15. Feedback Loop:

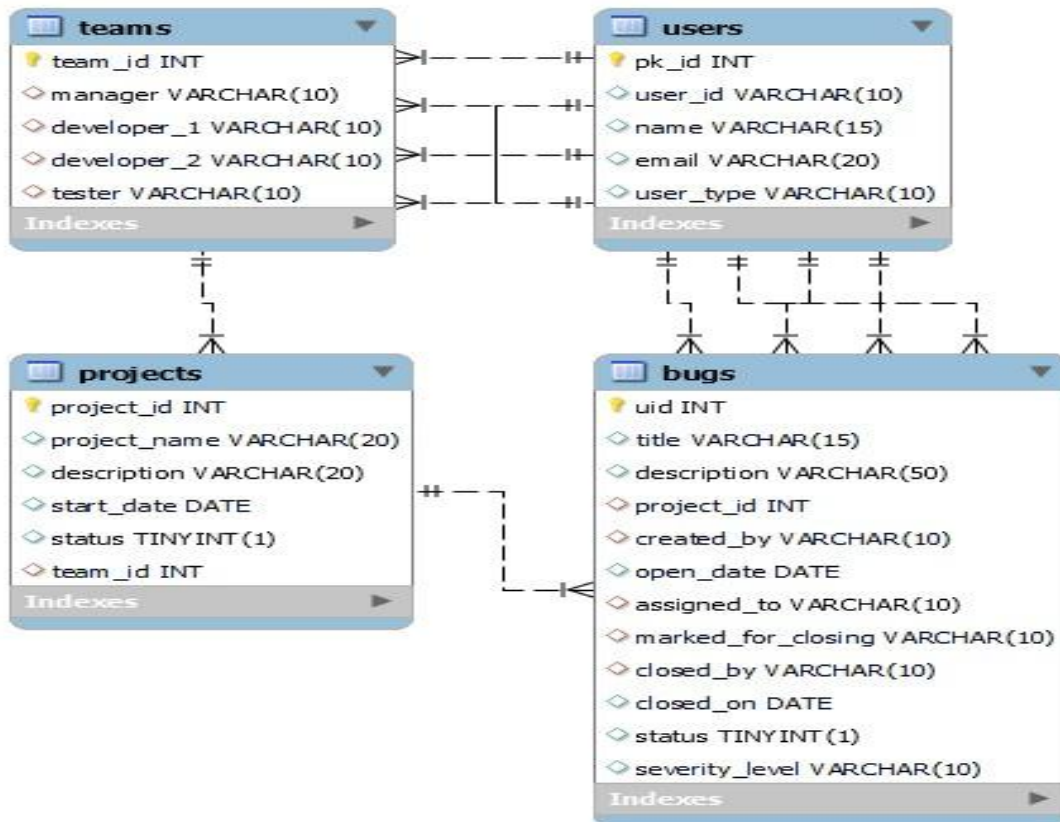
Feedback Gathering: Continuously gather feedback from users and stakeholders to drive further improvements.

16. Future Enhancements:

Feature Roadmap: Develop a roadmap for future enhancements and features based on evolving needs and technological advancements.

Diagrams

ER DIAGRAM



Relationships:

Team-Project Relationship:

- A Team can be associated with multiple Projects.
- A Project is associated with one or more Teams.
- This is a many-to-many relationship, typically implemented using a junction table or a foreign key in either the Team or Project entity.

Team-User Relationship:

- A Team may consist of multiple Users.
- A User may belong to one or more Teams.
- This is a many-to-many relationship, typically implemented using a junction table.

User-Bug Relationship:

- A User may be assigned to multiple Bugs.
- A Bug may have one assigned User.
- This is a one-to-many relationship, where each Bug has one assigned User, but a User may be assigned to many Bugs.

Project-Bug Relationship:

- A Project may have multiple Bugs.
- A Bug is associated with one Project.
- This is a one-to-many relationship, where each Project can have multiple Bugs.

Bug-Related Bug Relationship:

- A Bug may be linked to other related Bugs.
- This represents a hierarchical relationship between Bugs, where one Bug can be related to multiple other Bugs.

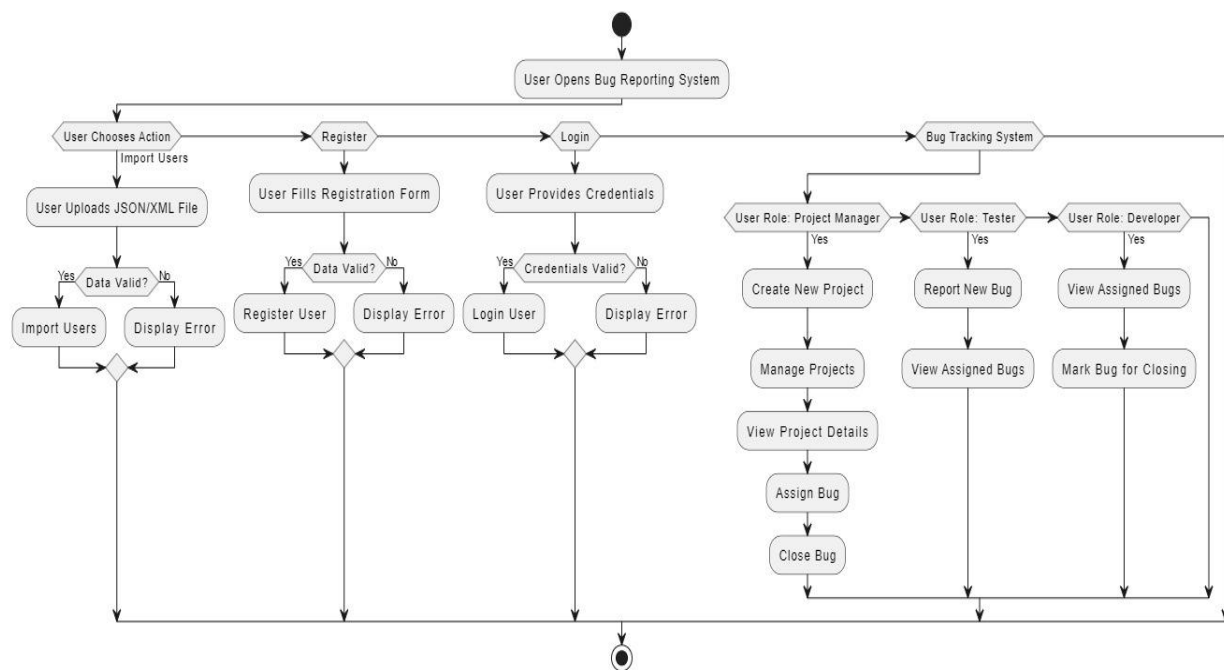
Attributes

- Each entity has attributes specific to its purpose. For example, Team entities have attributes like Team Name and Description, User entities have attributes like Username and Role, Project entities have attributes like Start Date and End Date, and Bug entities have attributes like Title and Severity.

Cardinality:

- The cardinality of each relationship (e.g., one-to-one, one-to-many, many-to-many) should be specified to indicate how entities are connected.


FLOWCHART



1. Start: The flowchart begins when a user interacts with the system.

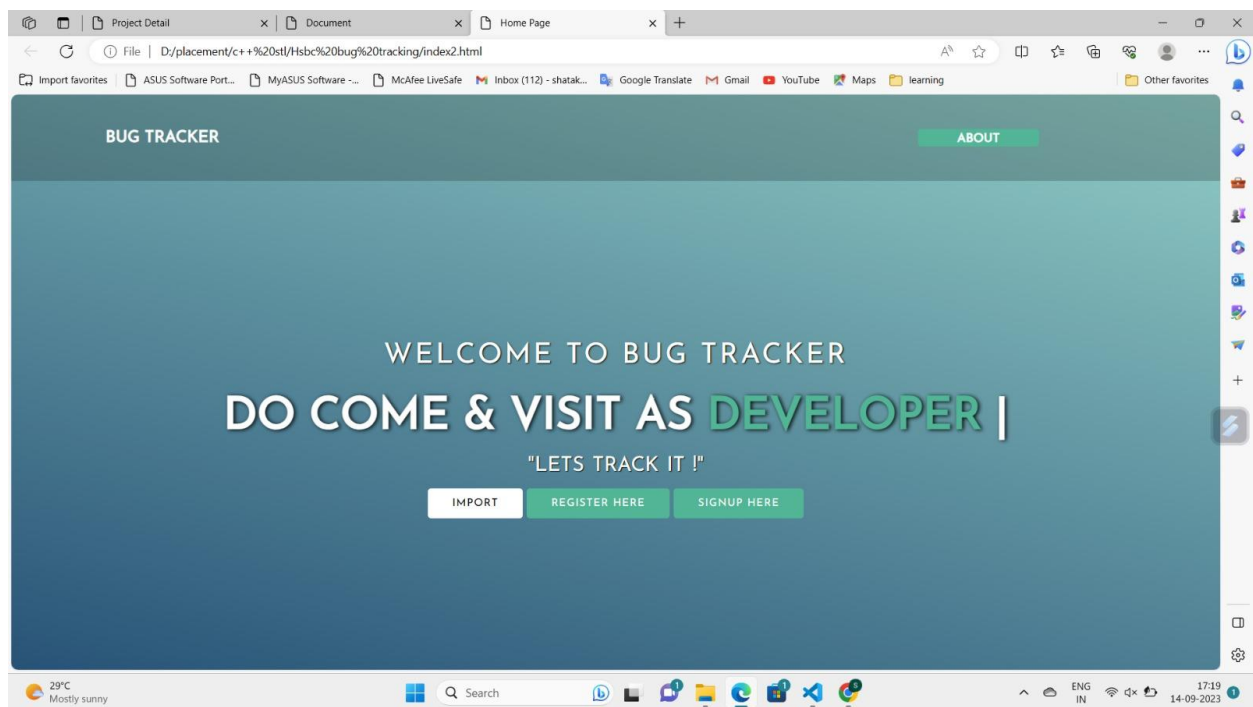
2. User Uploads JSON File?: This decision point checks if the user has uploaded a JSON file. If yes, it proceeds to the next step; otherwise, it moves to user registration.

3. Upload JSON File: If the user uploads a JSON file, the system processes the file, likely to import data (such as bug reports, project details, or user information) from the JSON file. After processing, it proceeds to the next step.

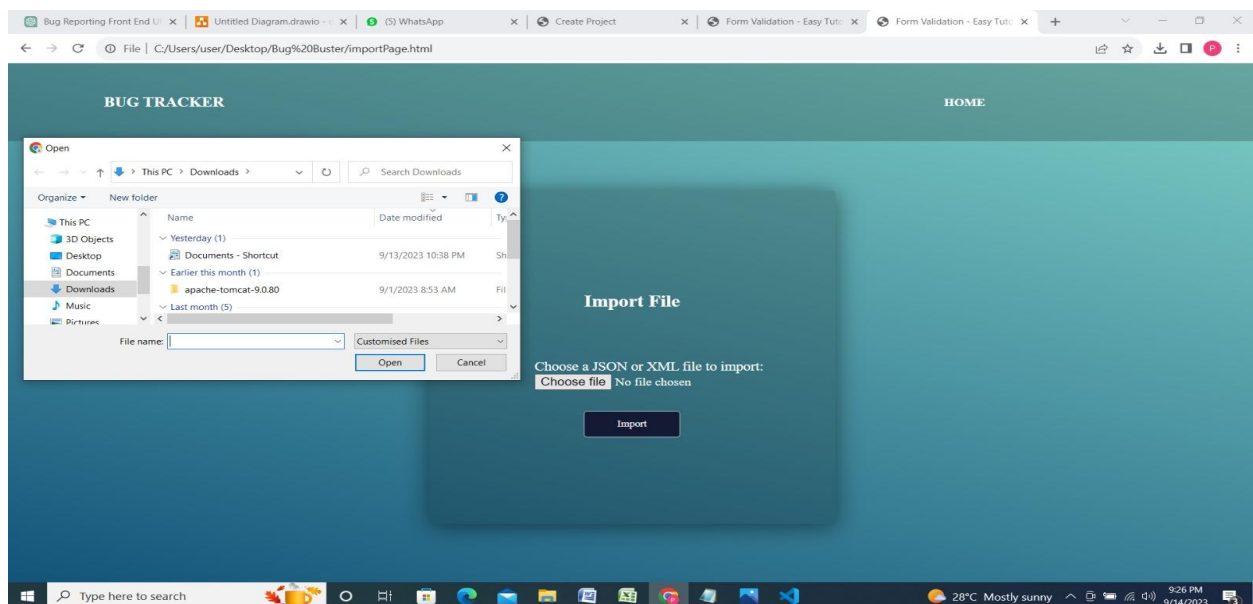
- 
4. **User Registration:** If the user did not upload a JSON file, they are directed to the user registration process. Here, they provide details like username, email, and password for registration.
 5. **User Registration Complete?:** After user registration, the system checks if the registration was successful. If registration fails (e.g., due to duplicate username or invalid email), the user is prompted to re-register. If registration is successful, the user proceeds to the next step.
 6. **User Login:** At this point, the registered user logs into the system using their credentials, including the username and password.
 7. **User Login Successful?:** The system verifies the user's login credentials. If login fails (e.g., due to incorrect username or password), the user is prompted to re-enter their login information. If login is successful, the user advances to the next step.
 8. **Choose Role:** After successful login, the user is presented with options to choose their role within the Bug Tracking System. Typically, users can select from roles such as manager, tester, or developer. The choice of role may determine their access rights and responsibilities within the system.
 9. **Role Selected:** The user selects their role, and the system assigns appropriate permissions and privileges based on that role.
 10. **Enter Bug Tracking System :** With their role determined and access rights configured, the user enters the Bug Tracking System, where they can perform tasks based on their role, such as reporting bugs, managing projects, or testing software.
 11. **End:** The flowchart concludes when the user enters the Bug Tracking System and begins using its features.

SnapShots

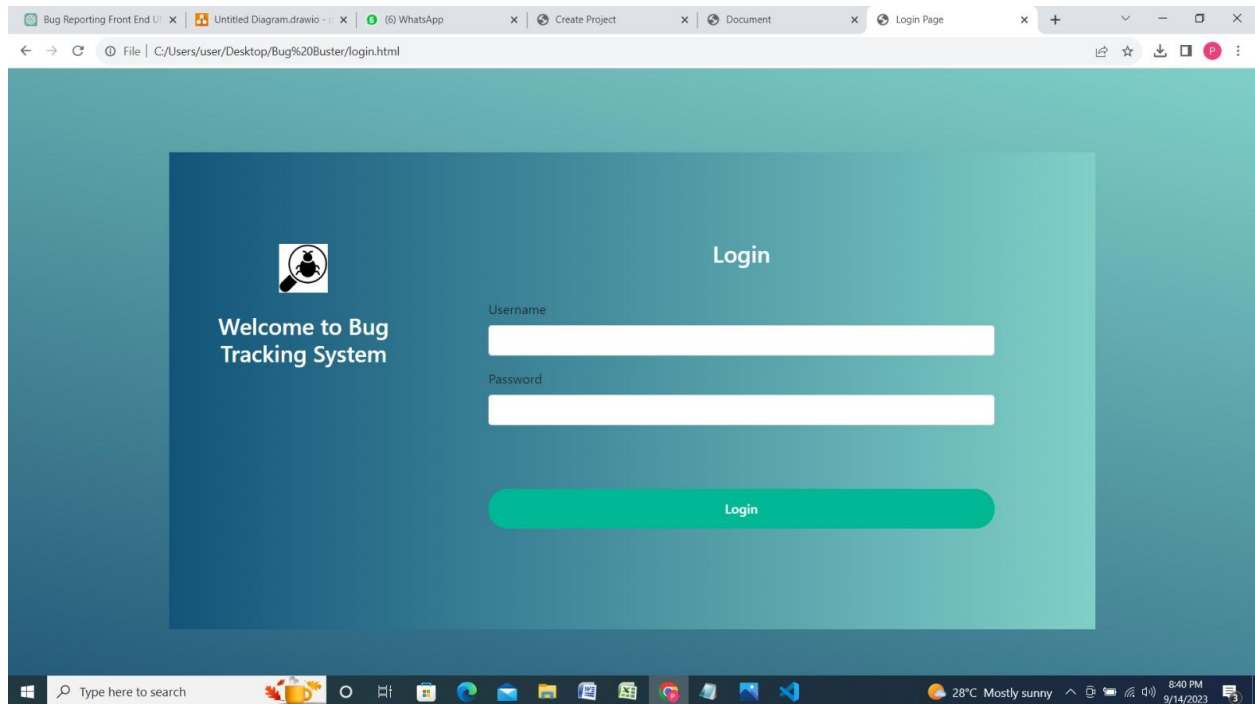
1. Home Page



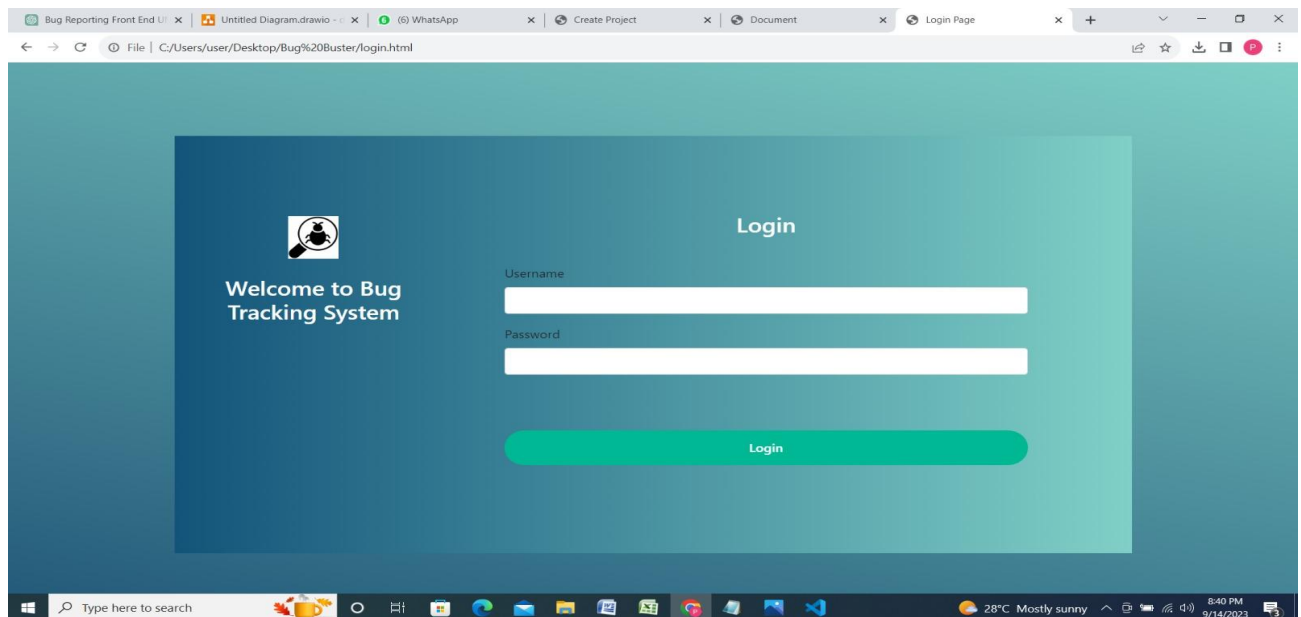
2. Import User Page



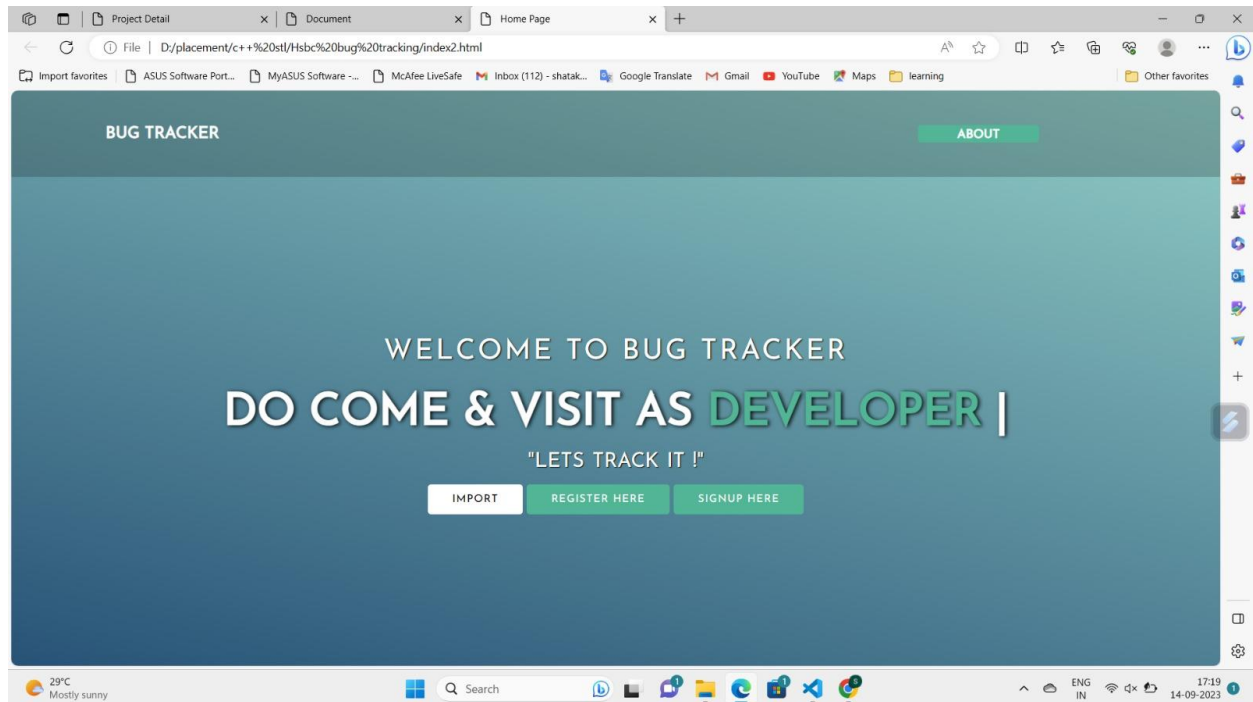
3. Register Page



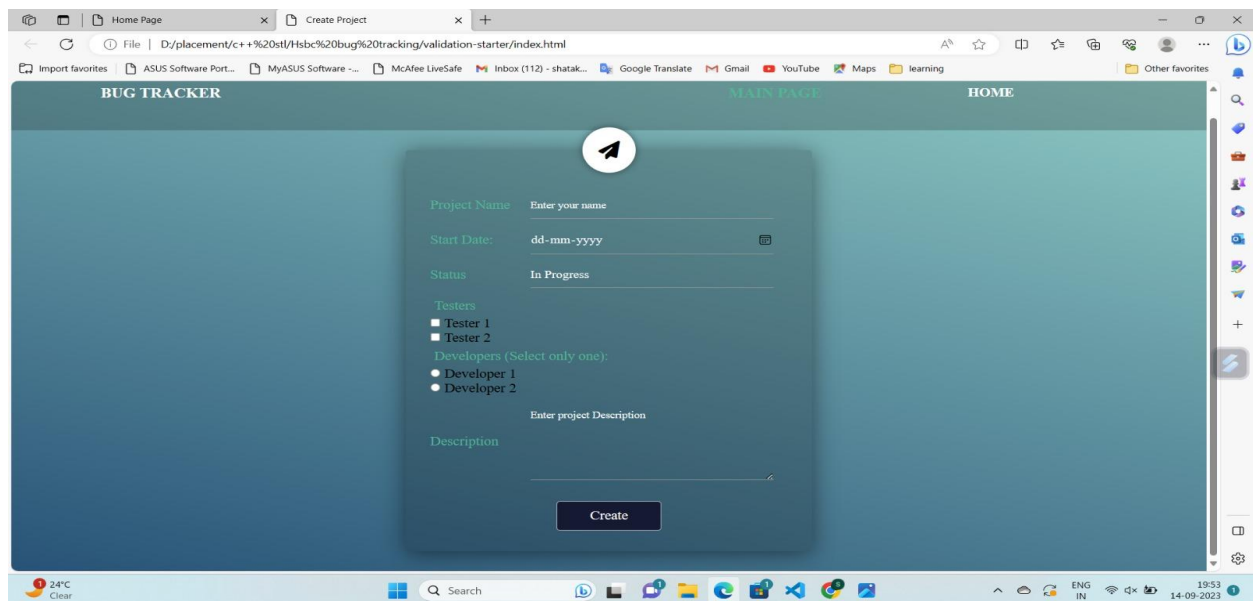
4. Login Page



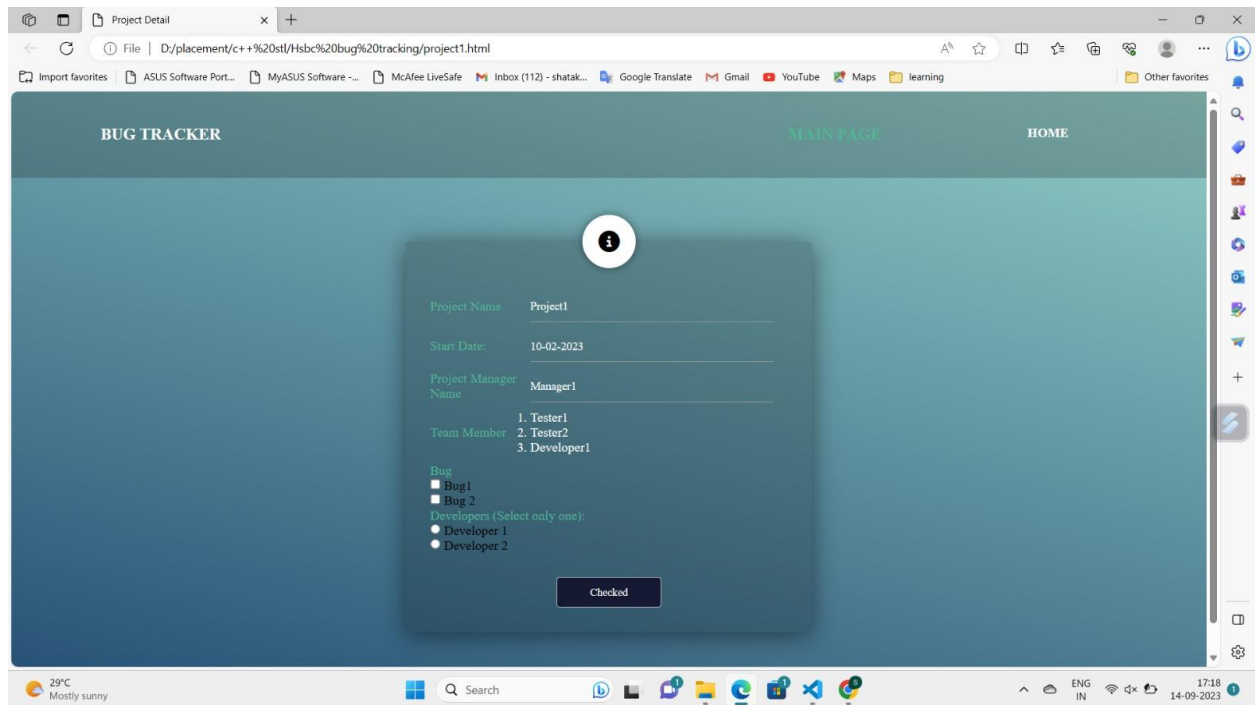
5. Main page



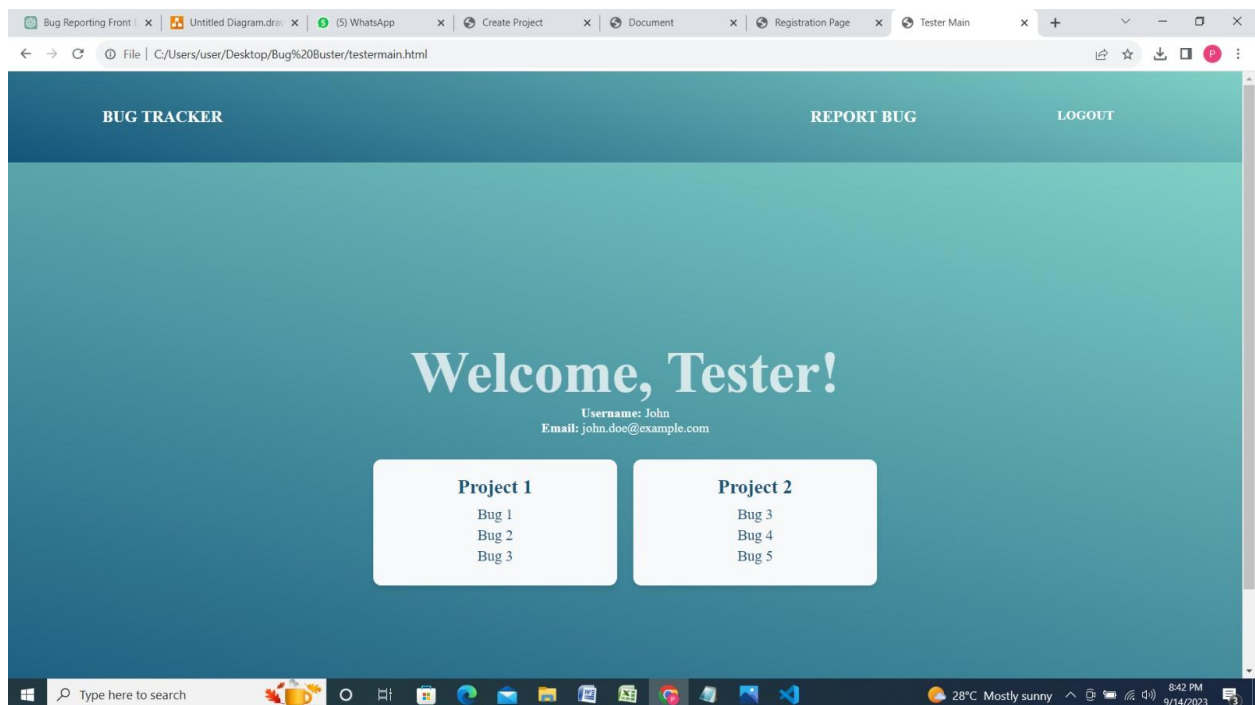
6. New Project Creation Project



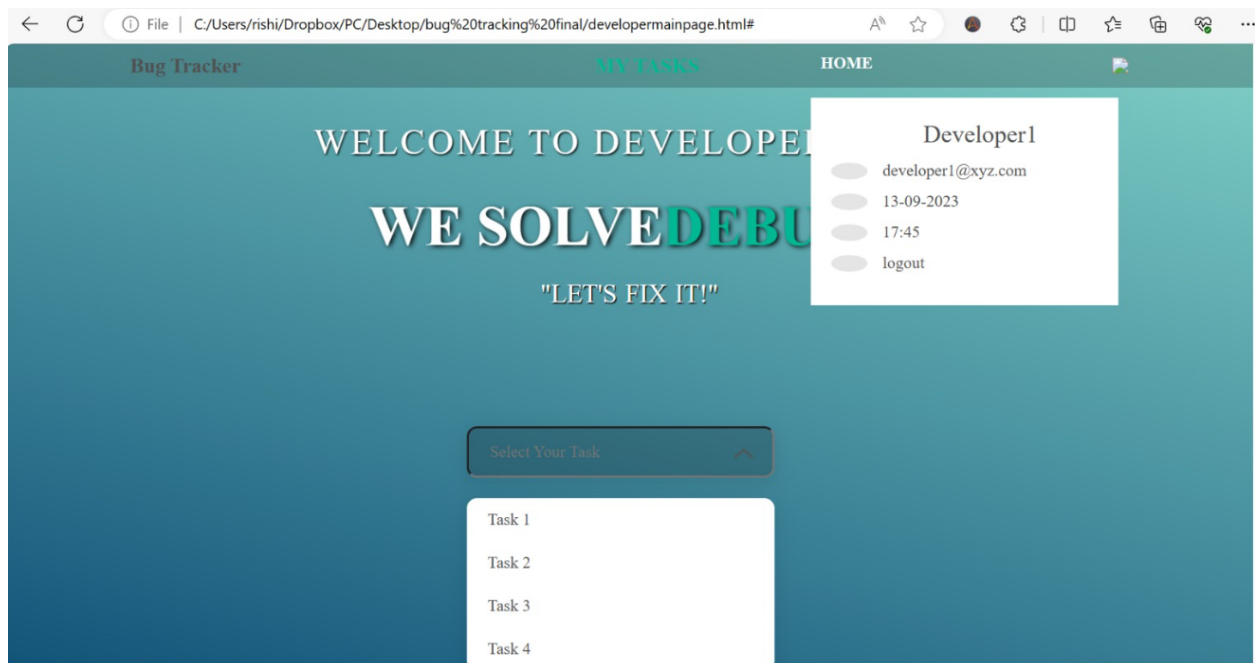
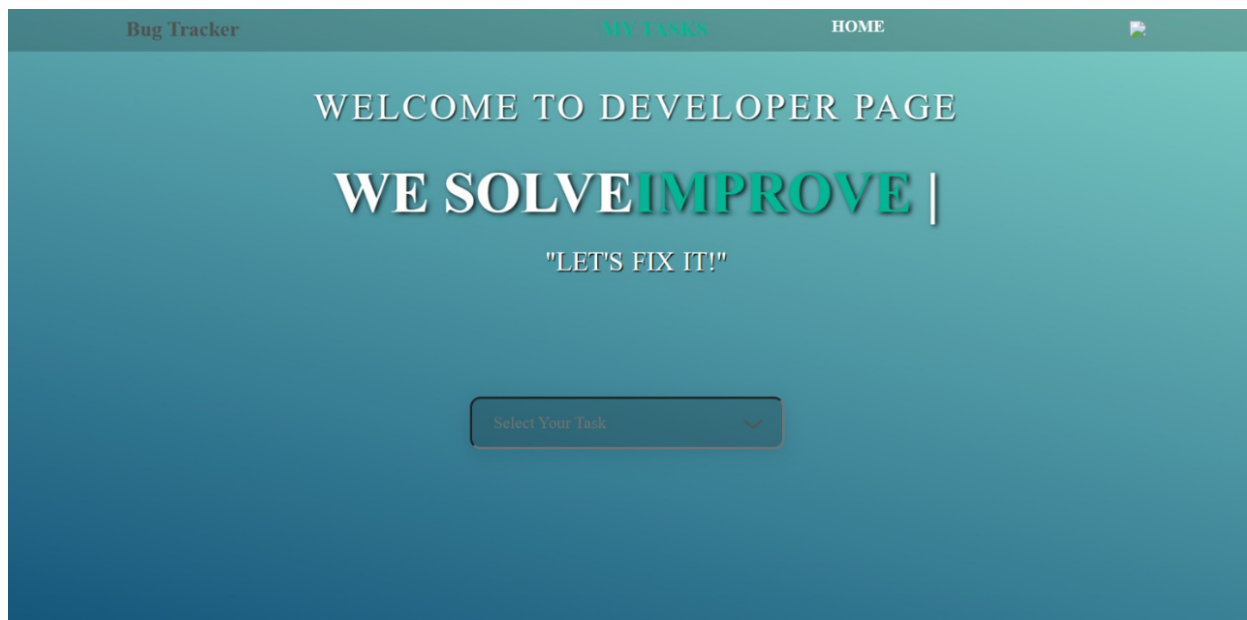
7. Project Detail Page



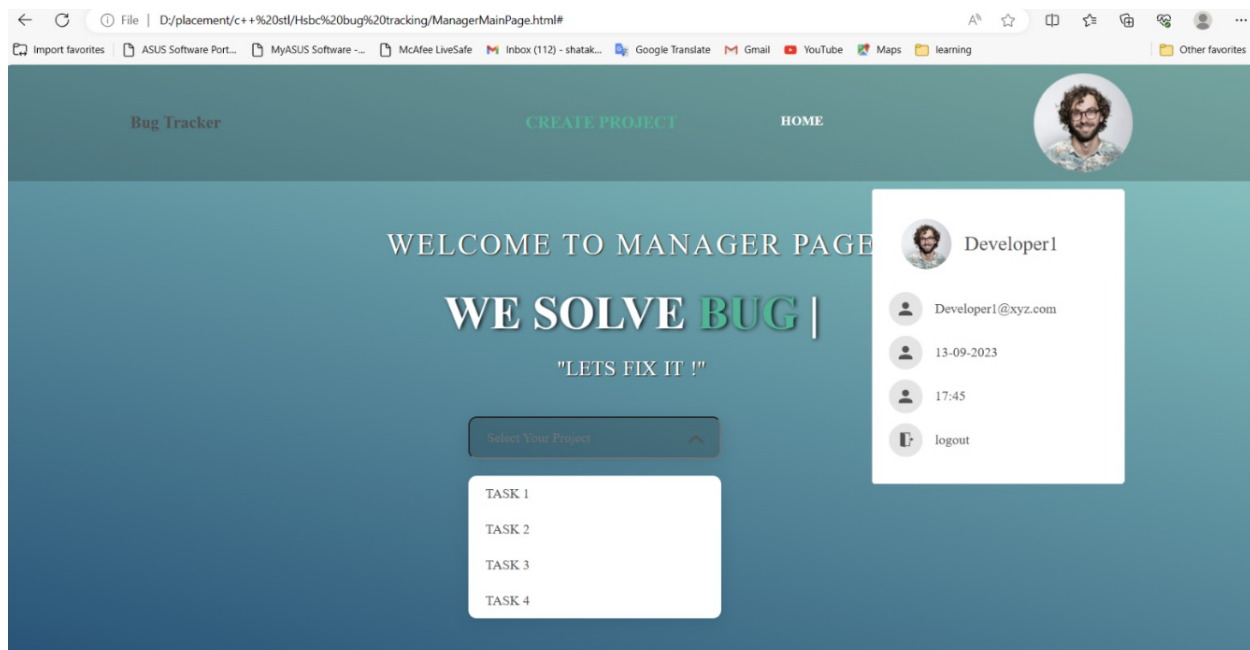
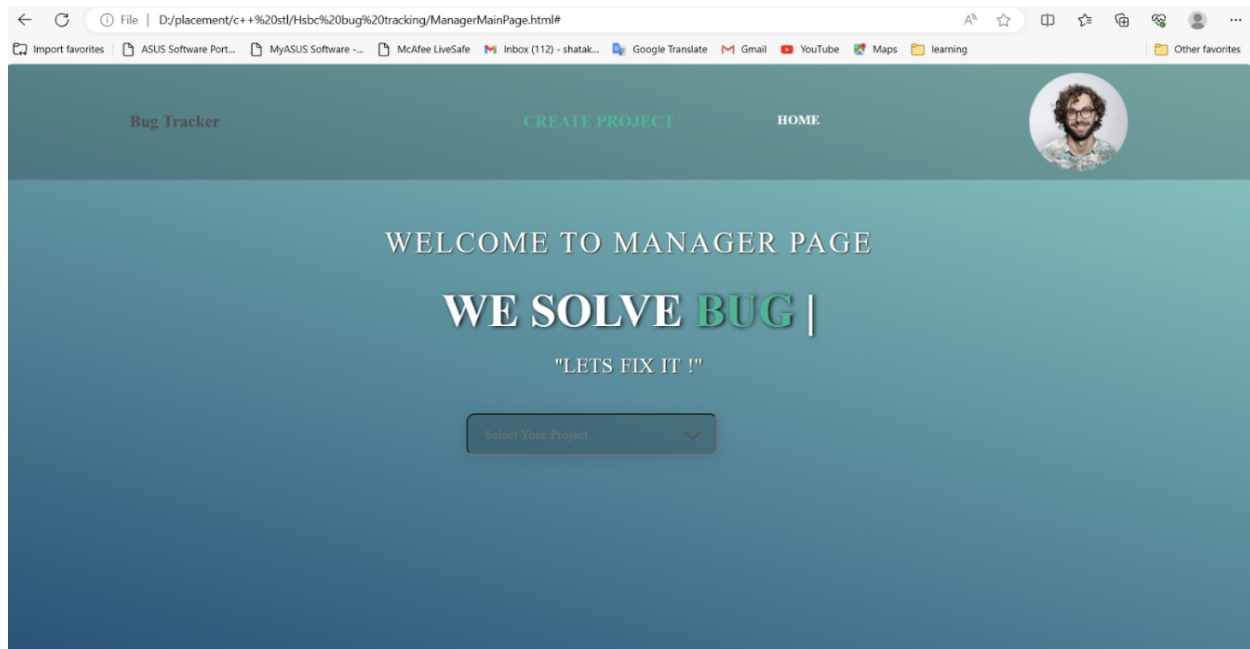
8. Tester Page



9. Developer Page



10. Manager Page



11. Report New Bug Page

BUG TRACKER HOME LOGOUT

Project Name: _____

Title: _____

Description: _____

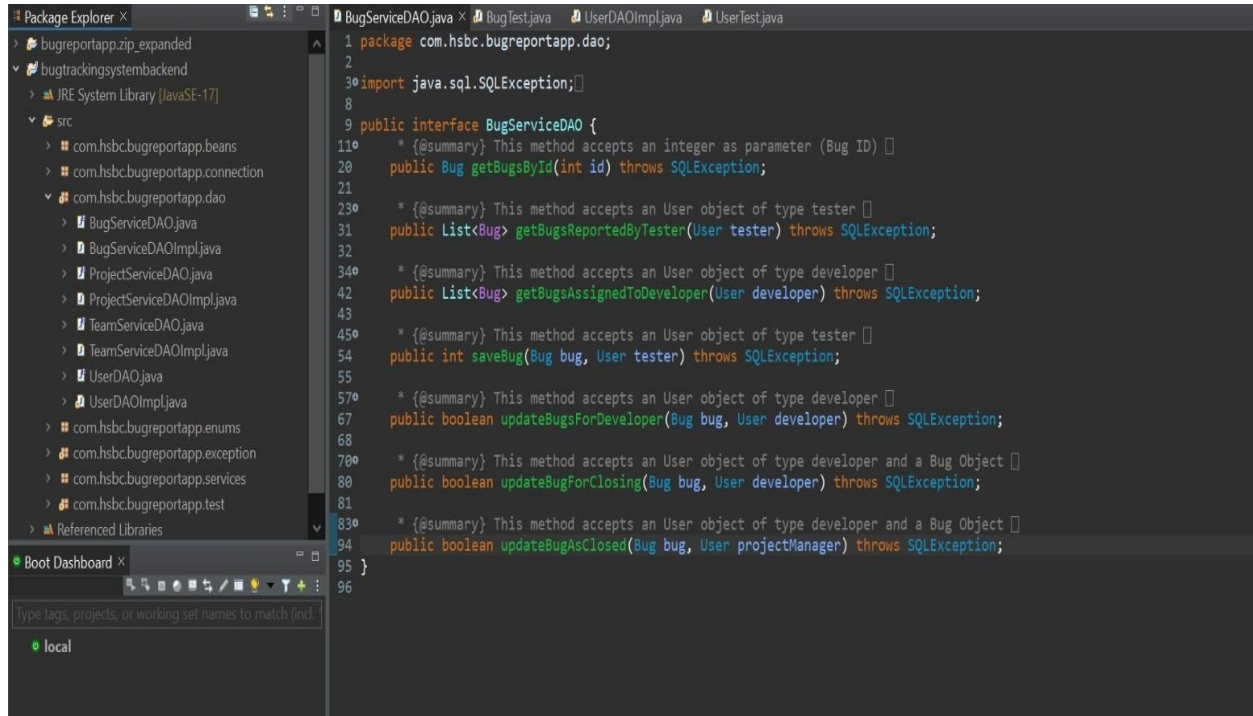
Severity Level: Low

Submit

Type here to search


28°C Mostly sunny 8:45 PM 9/14/2023

12. Layered Architecture



Future Scope

- Implement more advanced reporting and visualization tools to provide deeper insights into bug trends and project performance.
- Establish a feedback mechanism within the system to collect suggestions and feedback from users, driving continuous improvement.
- Adding SQL Injection Validator and Notification and Alert Feature
- Keep the system up-to-date with the latest security standards and best practices to safeguard sensitive data.

- 
- Explore the use of machine learning algorithms to automatically detect and categorize bugs based on historical data and patterns.
 - Implementing User Session Management feature.